Technical Information Center (TIC) Report Cover Page

Registration No.

23453

TARDEC

-Technical Report-

"Fuel Efficient Demonstrator (FED) Alpha In-dash Vehicle Display Transition to Apple iPad for Reduced SWAP-C"

OCT 2012

U.S. Army Tank Automotive Research, Development, and Engineering Center
Detroit Arsenal
Warren, Michigan 48397-5000

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 17-10-2012	**2. REPORT TYPE** Technical Report	**3. DATES COVERED** *(From - To)* June 2012 – December 2010
4. TITLE AND SUBTITLE Fuel Efficient Demonstrator (FED) Alpha In-dash Vehicle Display Transition to Apple iPad for Reduced SWAP-C		**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Richard Chase		**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army TARDEC - VEA, 6501 E 11 Mile Road, Warren MI 48397, MS224		**8. PERFORMING ORGANIZATION REPORT NUMBER** 23453
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army TARDEC, 6501 E 11 Mile Road, Warren MI 48397		**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** 23453

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper discusses the transition of the Fuel Efficient Demonstrator In-dash Vehicle Display (FED IDVD) from a ruggedized computer to an Apple iPad. The hardware differences between the original IDVD used in the vehicle and the iPad are discussed, as well as the differences in developing the software. A high level overview of the development of the iPad application is given, along with some code examples and comparisons with the original code as well as code examples compared to the original code. The reduced size, weight, power and cost (SWAP-C) are examined, and lessons learned from this project are discussed.

15. SUBJECT TERMS

Fuel Efficient Demonstrator (FED), Vehicle Touchscreen Display

16. SECURITY CLASSIFICATION OF: UNCLAS			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Richard Chase
a. REPORT UNCLAS	**b. ABSTRACT** UNCLAS	**c. THIS PAGE** UNCLAS	Public Release	7	**19b. TELEPHONE NUMBER** *(include area code)* 586-282-4889

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Fuel Efficient Demonstrator (FED) Alpha In-dash Vehicle Display Transition to Apple iPad for Reduced SWAP-C

Abstract

This paper discusses the transition of the Fuel Efficient Demonstrator In-dash Vehicle Display (FED IDVD) from a ruggedized computer to an Apple iPad. The hardware differences between the original IDVD used in the vehicle and the iPad are discussed, as well as the differences in developing the software. A high level overview of the development of the iPad application is given, along with some code examples and comparisons with the original code as well as code examples compared to the original code. The reduced size, weight, power and cost (SWAP-C) are examined, and lessons learned from this project are discussed.

Introduction

The IDVD is a ruggedized computer used as a user interface for the FED Alpha vehicle [1]. The computer runs Windows XP and can therefore host a variety of applications. For this demonstrator vehicle, the IDVD acted as a multi-function touch screen similar to those found in commercial automotive applications. The IDVD development adopted design approaches from the automotive industry, and custom graphics were designed that can be embedded into other software and easily utilized on a variety of vehicles. Initially, the software development was done in LabVIEW, a system design software normally used in a laboratory environment. Eventually the software designed in LabVIEW was compiled to run on the IDVD by installing a LabVIEW real time kernel on the Windows XP operating system. Towards the end of the original IDVD project, Apple introduced the iPad, a tablet PC, and an experiment was done to test the feasibility of using an iPad in a military vehicle. The wireless capabilities of the iPad allowed it to function as in In-dash / Out-of-dash Vehicle Display (IODVD). The code and graphics work originally done in LabVIEW was ported to the iPad and tested. The following is a comparison of each method of developing a GUI in the respective hardware environment.

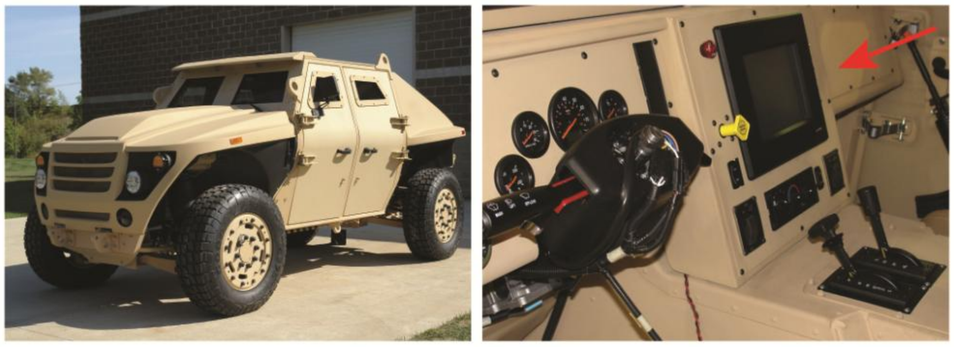


Figure 1: Exterior of the FED Alpha (left), and the In-dash Vehicle Display (right)

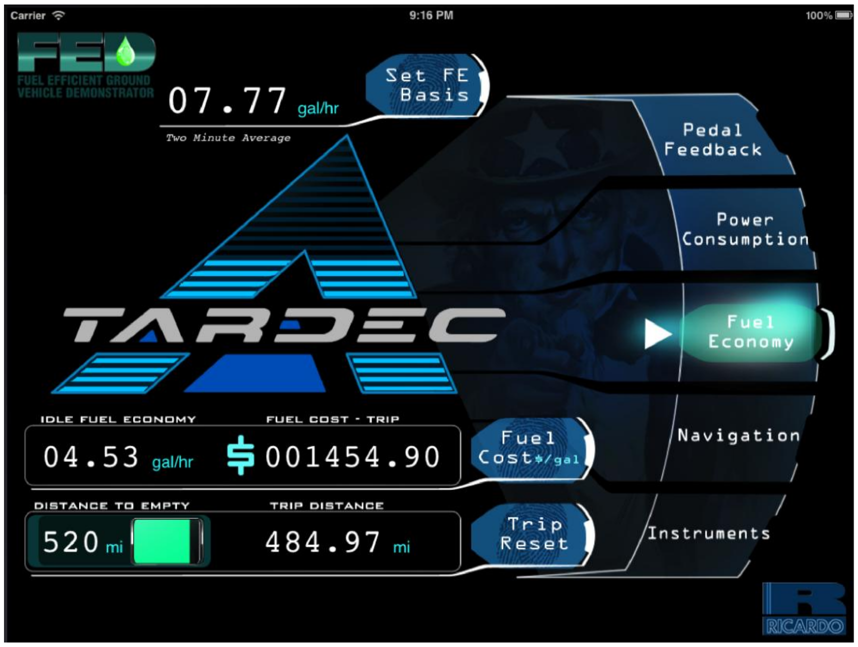


Figure 2: Screenshot of Fuel Economy Screen of IDVD GUI

VarTech Ruggedized Computer Features

The IDVD consisted of a VarTech brand ruggedized computer within a NEMA enclosure (VARTEC VTPC104PHB). Optional additions to this system are a CAN bus, keyboard and mouse interface, USB, RS232, etc. Many of the interfaces that come on the standard model were not used or needed in the final product. Furthermore, the particular unit used did not have a CAN bus directly on it, and a USB to CAN converter had to be used. The unit consumes 60 Watts regardless of the CPU usage, and it also has a long boot up time of approximately four minutes. The screen is also a resistive touch screen, and it does have an on screen keyboard and mouse. However, a physical keyboard mouse was attached due to the screen's slow and cumbersome operation. At the time, a resistive touch screen was chosen as many war-fighters wear gloves, and gloves inhibit the operation of capacitive touch screens.

The system ran the Windows XP operating system, which allows many different types of applications to be run on the system. LabVIEW was used as a test bench to debug the system on, and the front end GUI was made from the existing LabVIEW modules. However, the LabVIEW development suite has limited GUI capabilities. Coding in the LabVIEW environment is done graphically, and although there are some elements of GUI objects that can be controlled, there is not an extensive list of elements. For example, one may be able to control the background color of a window, but not the window border shape and size. The library must be built into LabVIEW natively, which means that the developer can only control the functions that are already built into the system. If the developer wishes to control a GUI element that does not have a native function assigned to it, a workaround must be created which adds development time and normally makes the program run slower. However, since the operating system is Windows XP, many different types of development environments can be used to create GUIs that will run on the IDVD. LabVIEW was chosen as the method for creating the GUI because that backend was developed in the lab and had undergone significant testing. Although the GUI ran slow in the LabVIEW environment, it ran smoothly when compiled.

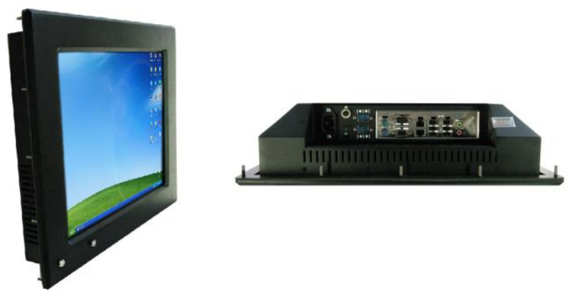


Figure 3: Multiple angle view of VarTech ruggedized computer

iPad Features

The iPad was released towards the end of the IDVD development, and it was decided to experiment with creating the FED GUI on it. The iPad requires significantly less power and also had wireless capability, allowing its use inside and outside of the vehicle. A proprietary port on the bottom of the iPad provides the capability to plug directly into the CAN bus through a middleware bridge. It can also be charged and powered directly from the middleware bridge. The iPad uses a capacitive touch screen which is much more responsive than a resistive touch screen, although the finger tips of a war-fighter's gloves must be modified in order to interact with a capacitive touch screen. Another key advantage of using an iPad is that the development suite, called Xcode, is specifically designed for GUI creation, and the integrated libraries make use of the native hardware which means that programs run extremely fast. On top of that, if the developer needs to manipulate a GUI widget in a manner that is not specifically designed into the native libraries, new functions can be easily added with a few lines of code resulting in highly customizable graphic displays.

Figure 4: Side view of Apple iPad

Comparison of Coding Methods

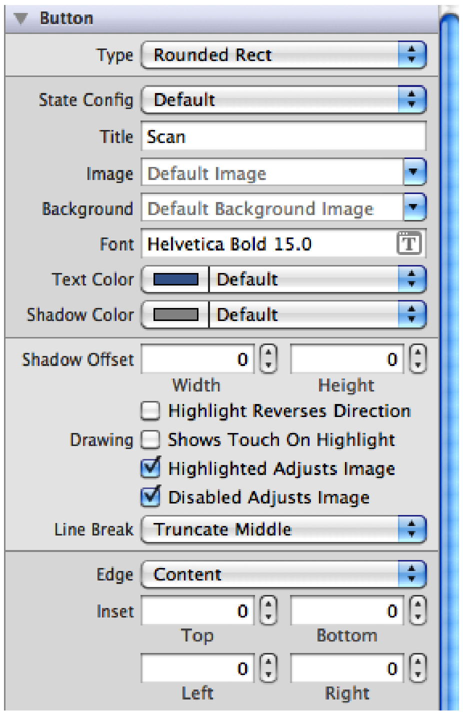
GUIs for iOS are written in Objective C, a modified form of C++. There is a small learning curve when beginning to use Objective C for those coders that have a C++ background, but the language is fairly intuitive and it allows for coders to understand other coders work much easier than C++. GUIs are written mainly in code as opposed to the graphical coding environment of LabVIEW. Libraries are also available to control almost every aspect of a widget, which means that a GUI's look is highly customizable. For quick start up of an application there is also an application called Interface builder that can be used alongside Xcode. Interface builder allows the developer to graphically create a simple GUI window without coding. These windows can then be called into the parent window, and opened and closed with function calls. The difference between this method and LabVIEW is that interface builder allows the coder to access the GUI at a text based code level as well: LabVIEW does not. One drawback of using Xcode and objective C is that iOS devices only run signed code, which means that application must be provisioned and licensed to run on an iOS device. This can be completed by either uploading the app to Apple store and making it available for download, or by obtaining a developer's license which allows applications to be provisioned upon distribution.

Specific Examples of Coding Differences

Let us look at the specific example of defining a button state, and its coinciding images. For this example, a button will have three states – on, off, depressed. The depressed state of the button is when a finger is actually touching the button. Let us compare the two methods, first in Xcode for iOS, second for LabVIEW.

In Xcode there are 2 ways to define characteristics of the button. First is through coding. The button class has many functions such as setting the button type, title, color for each individual state, title for each individual state, etc. These specific calls can be looked at in the documentation folder of Xcode. This document is meant to be a high level overview of the two processes.

The second method is through the Xcode development environment. There is a properties window in interface builder that allows you to control most aspects of a button at each particular state: title, font, color, shadows, etc.



Button

Type: Rounded Rect

State Config: Default

Title: Scan

Image: Default Image

Background: Default Background Image

Font: Helvetica Bold 15.0

Text Color: Default

Shadow Color: Default

Shadow Offset: 0 (Width) 0 (Height)

Drawing: ☐ Highlight Reverses Direction, ☐ Shows Touch On Highlight, ✓ Highlighted Adjusts Image, ✓ Disabled Adjusts Image

Line Break: Truncate Middle

Edge: Content

Inset: 0 (Top) 0 (Bottom) 0 (Left) 0 (Right)

Figure 5: Screenshot of property editor in Xcode

In LabVIEW, it is difficult to define a button that has different images for an off and on position, but it is possible. Figure 4shows the base process for customizing a button in LabVIEW. First the coder must go to an advanced customization pane; second he must select the button mode and then edit function. Once that is done, they must select the picture to import. LabVIEW deals in predetermined images?absolute pictures, meaning that the picture must be designed beforehand, and LabVIEW simply either shows or hides the photo. To change the button image for a different state, this process must be repeated again. This works for both the ‘on’ and ‘off’ button state, but it is not possible to select the ‘depressed’ button state, meaning that the button will revert back to its default image when it is being pressed . This is undesirable when developing a sleek GUI application.

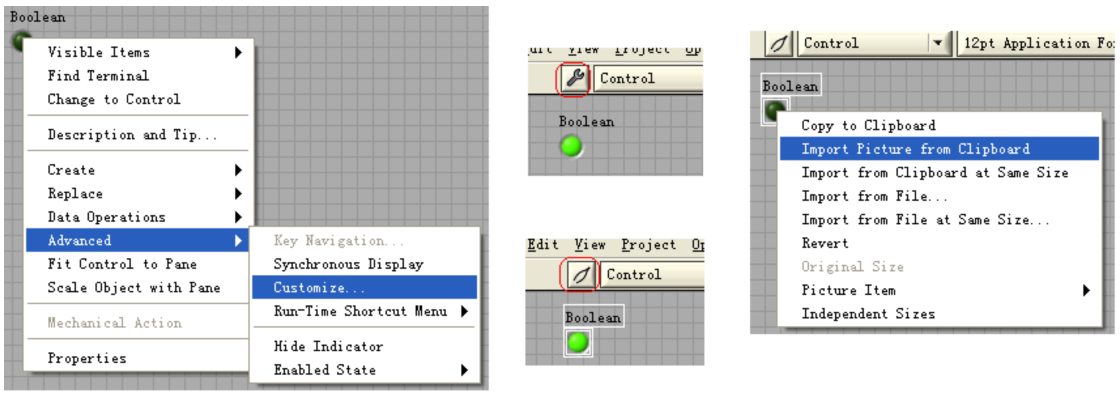


Figure 6: Screen shot of property editor process in LabVIEW

The interface from the iOS version of the IDVD to vehicle's CAN bus was done through the use of a simple wireless to CAN bridge. The bridge was able to read CAN bus packets without interfering with vehicle's CAN bus, by simply reading in messages and not transmitting packets. . The unit does have the ability to transmit CAN messages on the bus, but this feature was turned off for safety reasons. The vehicle had multiple CAN busses, and the one that the iPad was attached to also had the engine, transmission, and other critical components on it.

Comparison of VarTech Ruggedized Computer and iPad

Table 1: Comparison table of iPad and VarTech performance

	VarTech	iPad v1	SWAP-C
Sunlight readable	Yes	Hard to Read	**Loss in Function**
Input voltage range	9 to 36V	Rechargeable Battery Power (10 hrs, 1 month in standby)	**Gain in Function**
Resolution	1024 x 768	1024 x 768	Even
NEMA Enclosure	NEMA 4 (IP65)	No	**Loss in function**
Operating System	Windows XP	iOS 5.1.1	
Boot up time	4 Minutes	Instantaneous from Standby Mode	**Gain in Performance**
Power consumption	60 Watts Continuous	4.2 Watts @ Full CPU load	**>93% reduction**
Temp range	-20 to 70 C	0 to 35 C	**55 Degree loss**
Dimensions	20 x 17 x10 in	9.56×7.47×0.528 in	**98% Reduction**
Weight	13 lbs	1.6 lbs	**88% Reduction**
Processor	T4500 2.30 GHz	1 GHz ARM Cortex-A8	Loss (Acceptable)
Memory	2 GB	256 MB	Loss (Acceptable)
Sensors	None	Accelerometer, Ambient Light, Magnetometer, GPS	**Gain in functionality**
Network	10/100BaseT	Wireless (802.11 a/b/g/n), Bluetooth, Cellular	Loss / Gain in functionality (Even)
Display Size	10.4 in	9.7 in	**14% Loss in Area**
Touch Screen	Resistive	Capacitive	Even
Cost	$5,300	$400	**93% Reduction**
I/O	Varies- CAN, USB, etc	Proprietary I/O connector	Even

The iPad is not a ruggedized piece of equipment and there is some functionality loss associated with that. The display is not easily readable in the sunlight, it does not have a weatherproof NEMA enclosure, and its operating temperature is not suitable for military environments yet. However, with some manufacturing adjustments, the device could be built to withstand harsher environments. There is also a 14% loss in screen size.

Some features between the two come out even. For example, the screen resolutions are the same. It is important to note that new versions of the iPad released during the writing of this document have much higher screen resolutions. The processor and memory of the iPad are less powerful than the VarTech PC, but with the iOS code taking advantage of specific hardware, similar performance was realized with decreased computing power. Another even trade off is the I/O and network connections. While the iPad does not have a physical connection for network ports, it has wireless capable. Also, the full sized computer can have CAN and USB connectors built into it with additional cost, the iPad has a proprietary serial connection that can attach to an add on board which can easily be developed.

In terms of SWAP-C the iPad outperforms the built in computer easily. It has a 10hr battery lifetime and can boot instantaneously from standby mode. It also has a 93% reduction in power, 98% reduction in volume, 88% reduction in weight, and a 93% reduction in cost.

Lessons Learned and Conclusion

Each version of the IDVD had its pros and cons associated with it. There was upfront design time associate with each, but both were equal. LabVIEW and the iOS platform each have graphical interfaces to build basic items which allow them to have a working baseline up and running quickly. However, with LabVIEW, there was a significant number of workarounds that had to be developed to customize the interface to make it look sleek, whereas iOS and Xcode has the ability to customize virtually every aspect of the interface. The capacitive touch screen on the iPad was much more responsive than the resistive touch screen on the VarTech ruggedized computer. With ruggedization, the iPad could outperform any full size ruggedized computer that is used for an IDVD.

Another issue to consider is the licensing agreement between the government and Apple required in order to provision programs to the iPad. Both entities have their own set of bureaucratic processes, and navigating through the legal processes of each resulted in a 6 month long acquisition for a $300 purchase. We were able to develop code during the acquisition process so no time was lost. Now that the acquisition is finalized, TARDEC now has a license to provision apps for in-house applications such as demonstration vehicles, info apps, and testing in labs.

References

[1] R. Chase, “Development of the In-Dash Vehicle Display for the Fuel Efficient Demonstrator (FED) Alpha Vehicle,” U.S. Army TARDEC, Tech Rep. 2012